Ontology-based methodology for *e*-Service discovery

Devis Bianchini [a,*] Valeria De Antonellis [a] Barbara Pernici [b]
Pierluigi Plebani [b]

[a] *Dipartimento di Elettronica per l'Automazione, Università degli Studi di Brescia*
Via Branze, 38 - 25123 Brescia, Italy

[b] *Dipartimento di Elettronica ed Informazione, Politecnico di Milano*
Piazza Leonardo da Vinci, 32 - 20133 Milano, Italy

Abstract

Service discovery is a critical aspect in the Service Oriented Computing approach. A model, a methodology and a tool environment based on ontologies are proposed in this paper. The requester and provider perspectives are discussed, both to support the service publication phase and the search phase. The proposed service ontology is based on functional aspects and it is organized on three layers, to support traditional search based on classification such as proposed in UDDI as well as search based on abstracting service characteristics. In addition, non functional features such as requester and provider contexts and quality of service are considered to refine the search results according to the requester requirements.

1 Introduction

Service Oriented Computing (SOC) is a new emergent paradigm on which service technologies rely. According to [1], Service Oriented Computing is the computing paradigm that uses services as fundamental elements for developing applications. The paradigm is based on the service provider and the service requester as the main actors involved, where the former, on the basis of his mission and available resources, builds a set of invocable software applications that provide some functionalities; the latter, on the basis of the service description provided, invokes and uses such software either in an interactive way or building suitable service software clients.

* Tel. (+39) 030 3715447; fax: (+39) 030 380014
Email addresses: bianchin@ing.unibs.it (Devis Bianchini), deantone@ing.unibs.it (Valeria De Antonellis), barbara.pernici@polimi.it (Barbara Pernici), pierluigi.plebani@polimi.it (Pierluigi Plebani).

As it occurs in real life, in which traditional services are available everywhere (e.g., flight booking, stock option reservations, phone calls and so on), in a virtual service-based environment mechanisms to easily find the interesting services are absolutely necessary for both the service provider and the service requester. On the one hand, from the service provider standpoint, the higher the service reachability, the higher the probability that someone will use it. On the other hand, the service requester needs to be able to find the more suitable services with respect to his requirements.

For these matters, the Service Oriented Architecture (SOA) relies on service descriptions and basic operations (publication, discovery, selection and binding) to enable the provider to make information available about his services and the requester to find the appropriate services and use them. Publication, discovery and selection are based on a Service Directory. The role of such an actor is to provide methods which (i) can be used by the service provider to publish the provided services and (ii) can be used by the service requester to look for the services which satisfy his requirements. UDDI Registry [2] is the well-known tool available which implements the service directory facilities as described above. However, even if UDDI allows for browsing the service registry through different indexes, it does not provide (i) an effective content-based service discovery, (ii) a match with respect to the context in which the service operates and (iii) a quality evaluation of the retrieved services. In fact, services are only organized with respect to pre-defined categories (e.g. UNSPSC, NAICS) and they are not analyzed and described with respect to what they effectively provide.

To improve service description, the Semantic Web [3] research area provides a set of interesting methods and tools to augment service description based on the use of ontologies. As stated by [4], "the idea behind the Semantic Web is that generators of Web pages or services will create formal declarative descriptions that programs can use to find the appropriate service and use it correctly". Therefore, the focus is on adding descriptions to services using descriptive logics in order to be able to reason about their properties. The problem of this approach is that it requires the providers to perform such annotations, on the basis of ontologies commonly agreed upon by wide communities.

The service-oriented approach has been proposed mainly in an architectural context in which providers and requesters use systems which are available most of the time. Here the interaction is mainly web-based and providers are registered in the registry regardless of their operating conditions. However, many current applications are developed in highly variable environment, where mobile devices are becoming an integral part of current information systems, access to services may depend on the context in which the services are used and cooperative enterprises combine dynamically available services selecting the best offers in a given moment.

These applications pose new requirements to the classical service-oriented approach, in particular: (i) the ability of both *abstracting* service character-

istics from their operating environment and in some cases selecting services on the basis of their *contextual features*; (ii) the possibility of selecting services both through interactive interfaces and through sophisticated matching algorithms.

The aim of the present work, supported by the MAIS Project [1], is to present how the use of an ontology-based approach can support service publication and discovery. We propose an approach in which an extended service description is used as a basis for providing service retrieval and publication facilities in an enhanced UDDI Registry. The ontology-based representation of services is proposed extending UDDI registration and retrieval functionalities, providing a semi-automatic organization of services in ontological levels. Therefore, instead of associating semantic information directly to services, semantic information is extracted from published services on the basis of a domain ontology and used as a basis to provide advanced searching functionalities. In particular, our solution starts from UDDI Registry and improves its functionalities providing a service model and a set of methods which allow a content-based discovery. Moreover, this work concentrates not only on Web Services, but on *e*-Services in general: we assume the use of services in multi-channel information systems, in which the same service could be provided and used through several channels, with different networks, devices and context characteristics. Therefore, the service retrieval functionalities proposed in this paper focus not only on functional aspects, but also on the context of invocation and the requester profile.

The paper is organized as follows: in Section 2, we present the *e*-Service model; in Section 3, we propose a three-layer *e*-Service ontology architecture; in Section 4, *e*-Service discovery based on the proposed ontological framework is discussed and in Section 5 the architecture underlying our work is presented. Finally, related work is discussed in Section 6, while Section 7 concludes the paper and outlines future research directions.

2 *E*-Service model

The MAIS Project analyzes the services in the context of multichannel information systems development [5]. In such systems, services may be invoked by a variety of different users, using different devices with possibly wireless connections, including not only portable computers, but also PDAs and mobile phones. In this way, the typical Web Service, where the Web represents the communication channel, is now substituted by the more general *e*-Service, where the same application logic can be exposed through several channels. In such mobile information systems, even if the overall executing process is predefined, the set of active users, as well as the *e*-Services invoked, may change. As a running example, in the rest of the paper we consider a typical

[1] Multi-channel Adaptive Information Systems.
Web site: `http://www.mais-project.it`

travel support process, where the end-user wants to reserve a hotel and a flight for a trip. In particular, we also want to handle, in an automatic or semi-automatic way, particular situations, like cancellation of a booked flight, which force the travel agency to reschedule both flights and other arrangements for its customers. It must also be able to provide such information timely to the involved users and service providers. In the present paper, we will focus mainly on issues related to service selection on the basis of given requests, with the ultimate goal of providing a flexible and highly adaptive service invocation environment.

The implementation of a system which realizes service discovery necessarily requires a complete and precise definition of *e*-Service. In our approach, *e*-Service description is addressed both from the service provider and the service requester perspective:

- *e-Service provider perspective*, which allows for specifying the functionality and contextual characteristics from the provider side;
- *e-Service requester perspective*, which allows for specifying the service request both in terms of requested functionalities and of requester operating environment and preferences.

Providing both perspectives enables to characterize in a specific way the requester requirements, which may be different from the *e*-Service characteristics as specified by the service provider.

Moreover, services may be provided through different channels, with different quality characteristics. In this situation, it is important to define not only the functionalities provided by the *e*-Services, but also the quality related aspects and the context in which the services are both made available and used.

2.1 E-Service provider perspective

From a service provider perspective, the model has to define, as shown in Figure 1, who offers the service (*Service Provider*), what the service performs (*Functional Description*) and on which *Channels* the service is available. Moreover, the service could be simple or composed by a set of services organized according to a composition framework [6–8].

The functional description is provided in terms of invocable *Operations* and exchanged *Input/Output parameters*, which may be described in a WSDL [9] specification.

To specify the observable *behavior* of the service, we specify the (partial) order in which the available operations can be invoked through pre- and post-conditions. Each operation is associated to a set of *PreConditions* and *Post-Conditions*, that predicate on the *Inputs* and *Outputs* parameters of the operation. The former must be verified before the execution of the operation, while the latter are satisfied after the execution. For instance, a `flight payment` operation could have a pre-condition "flight-reserved". Pre- and post-conditions

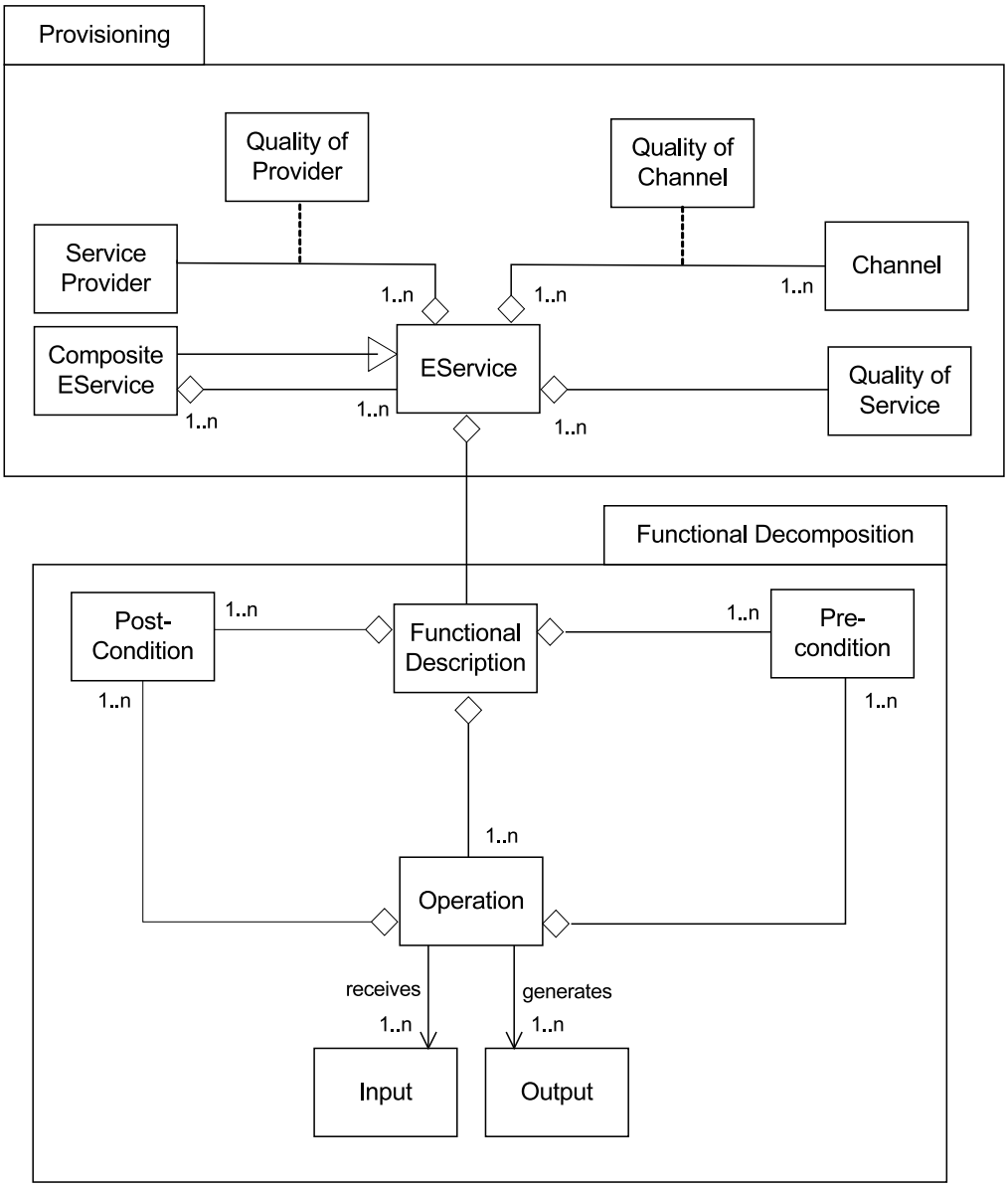

Fig. 1. Service provider perspective model.

can also be defined on whole services.

It is important to note that such a functional description relies only on the elements the provider identifies as "exportable" or, in other words, on the elements that should be made visible for a potential user to know in which way he can invoke the service. The implementation details are not represented by this model.

About non-functional aspects, each *e*-Service is characterized by a set of quality parameters. All the model is enriched considering the quality aspects. In particular, the channel, the provider and the service can be also detailed with respect to quality dimensions. It is worth noting that the involved quality dimensions strictly depend on the application and domain in which the

service, provider and channel operate. For this matter, since it is almost impossible to define a set of quality dimensions valid for all the possible services, we provide a more general solution. In particular, according to [10], we suppose that, for each kind of service, a community exists that defines which are the relevant aspects. Task of this community is to identify the set of relevant quality dimensions for such particular class of services, where a quality dimension is defined by the $\langle name, admissible\_value \rangle$ pair, where *name* represents a unique parameter identifier, whereas *admissible_value* represents the range of values suitable for the parameter. In this way, whoever wants to implement a service belonging to this class must refer to this set of quality dimensions in order to characterize its quality. In the same way, we suppose the existence of a set of communities, one for each kind of channel and a unique community which defines the quality of the provider (for us all the providers belong to the same class and we can use the same quality specification). Such an assumption reflects what occurs in real life, where, even if the community is not well defined, everybody agrees on a set of aspects to evaluate the quality of a given service. For example, considering an *e*-Service which provides an airport digital map, possible quality parameters could be: color-depth, resolution, width, length. On the contrary, if we consider a hotel reservation service, its quality could be evaluated with respect to the number and type of credit cards accepted to pay for the reservation. In this case, the range of admissible values is a discrete set composed, for example, by the list of the most accepted credit card companies.

In this way, our *e*-Service model includes on the provider side:

- a QoS (*Quality of Service*) definition, in which a set of admissible values is assigned to the quality dimensions identified by the related community;
- a QoP (*Quality of Provider*) definition, in which a set of admissible values is assigned to the quality dimensions defined by the Provider Community;
- a QoC (*Quality of Channel*) definition, for each channel available to invoke the *e*-Service, where a set of admissible values is assigned to the quality dimensions defined by the community of that particular channel.

A classification of possible general quality dimensions is being generated within the MAIS Project to represent qualities related to these classes [11].

2.2 E-Service requester perspective

¿From a service requester perspective, the model concentrates on the *User* definition, as shown in Figure 2. The requester of an *e*-Service is characterized by its *Profile* and by the *Contexts* in which the user can be and (by means of the *current* association) in which the user, in a given moment, operates.

The requested service is defined in terms of functional and non-functional aspects, as defined in the service provisioning perspective. The profile has a static and a dynamic component: the first one to render those properties that are statically set by the user (usually during the registration phase), the second

one to characterize all information that is collected while using the application. The static profile is defined by means of a set of user preferences such as *Role*, which identifies the role played by the user while using the application, its *Expertise* and *Ability* on the application and a set of *Generic Preferences* to add application-specific characterizations to the profile. The hypothesis is that roles or expertise define the minimum profile, which can always be enriched with further information: each *Generic Preference* has a name and value to let the designer render and "quantify" any property.

A definition of context is provided starting from previous work in the literature, such as UWA [12], and enriched with the channel definition. Here the context is defined by a *Location*, a *Time Zone* and the available *Channels*. *Location* can be zero or more *Geo Positions*, i.e., latitudes and longitudes, *Districts*, e.g., special-interest areas, *Towns* and *Countries*. Moreover, a *Location* can be associated with a set of *Properties*: a general-purpose mechanism to add further information to the context description. For instance, we could use a *Property* to specify weather conditions. *Time Zone* describes the *Context* with respect to its time information, i.e., its offset from Greenwich mean time, and the daylight saving time.

- *device*, defined according to a subset of the attributes included in CIM [13] model;
- *network*, considering an end-to-end link, we provide an abstraction of the several elements which are used to interconnect the provider with the user device (last mile excluded), characterized by the quality of service aspects;
- *network interface*, it represents the last mile of the connection and thus defines how the device could be connected to the network and how such a connection could influence the quality the user perceives;
- *application protocol*, it specifies the application protocol supported by the device according to a given network and network interface (e.g., HTTP, SOAP, SMTP).

As stated for the provider perspective, also the service requester specifies the non-functional aspects through the quality parameters defined by the service community. In particular the user can characterize the requested *e*-Service through only a subset of the quality parameters suggested by the service community considering only the interesting aspects. Using a hotel reservation service, a user could be interested only on the `creditcard` parameter and he requires that such a parameter must include at least one of the credit cards companies accepted. In this way, the user does not impose constraints on the other possible parameters, such as the cost of a possible reservation cancellation.

About quality definition, whereas in the provider perspective the model defines the quality offered, here the *e*-Service model identifies which are the needs of the user. Using the same approach described before, the user is aware about the existence of the communities and, in particular, about the service community. According to that, we assume that the user is able to define a precise

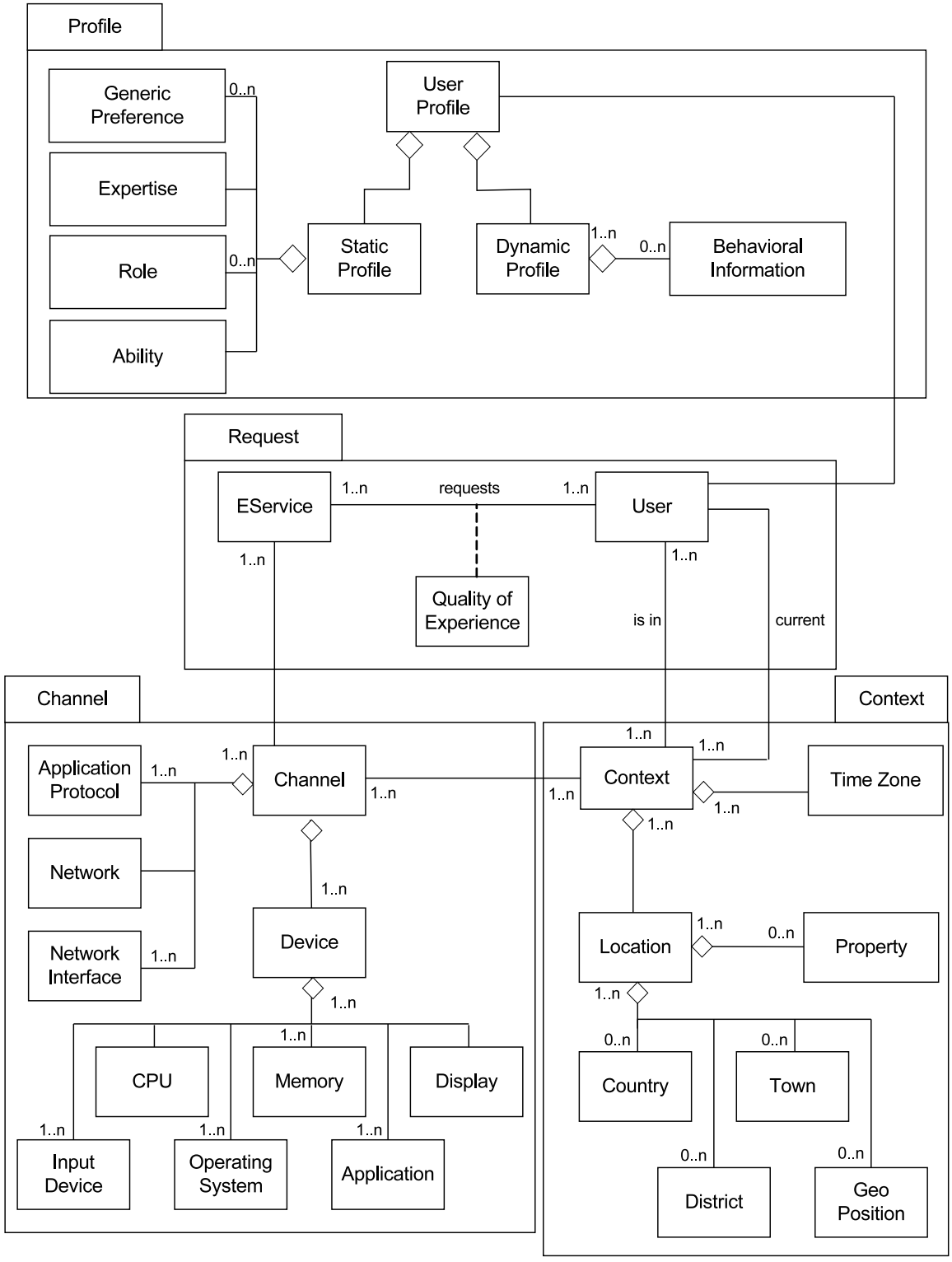

Fig. 2. Service requester perspective model.

QoS for each requested service, where for each quality dimension the admissible values represent the requested quality level. It is important to highlight that the service is connected to the user through different channels, which also influence the QoS. The network latency, for example, is different for a wireless connection rather than a wired one, so a typical QoS dimension (e.g., throughput) can be differently perceived by the user. For this matter the QoS required by the user cannot be directly matched with the QoS defined by the provider. According to that (and also according to the literature [14]) we introduce the Quality of Experience (QoE), where:

$$QoE = f(QoS, QoC, QoP)$$

In this way, for each triple $\langle service, channel, provider \rangle$ the user is able to effectively calculate which is the QoS that, during the invocation, the user can perceive. On the basis of the obtained values, the user will be able to define the admissible value of the quality of service dimensions for the service offered by a particular provider through a given channel. Even if the QoE is presented as a generic function, it is possible to find rules able to state the dependency among the different quality dimensions. For example, if the service provides streaming data, it is easy to state that the outgoing dataflow from the service cannot be greater than the bandwidth of the selected channel. In this way, simplifying and supposing that the service is able to provide an output flow of 1Gb/s, the user cannot obtain a flow rate greater than 100Mb/s if an Ethernet connection is used and 11Mb/s if a wireless LAN is used. Obviously, QoE=QoS holds whenever the quality parameters are not affected by the channel as, for instance, the credit card restriction described above.

3 Three-layer domain service ontology

In the previous section, we have introduced several elements to describe an *e*-Service: (i) the functional description to specify its operational aspects, (ii) the quality offered by the provider and requested by the user taking into account the involved channels, (iii) the user profiles and (iv) the context information that can be exploited to identify the particular conditions in which an *e*-Service is used. In the remainder of the paper we will show how a service ontology, that properly organizes *e*-Services on three different layers of abstraction, can be exploited to enhance service discovery on the basis of user functional requirements; further refinements of the discovery process can be performed by taking into account context information and, finally, selecting only those services that match user quality requirements.

In the service ontology, we introduce three distinct concepts, according to an increasing level of abstraction:

- *concrete services*, that are directly invocable services and that are featured by their public WSDL interface, which is used to group them on the basis of their functional similarity (as explained in the following) and by bindings

to specific implementations;

- *abstract services*, that are not directly invocable services, but represent the functionalities of sets of similar concrete services (each abstract service is associated to a corresponding cluster of similar concrete services); abstract service functionalities are also described by means of a WSDL interface and are obtained from the concrete service operations by means of an integration process; mapping rules are maintained among the abstract operations and the original concrete operations; abstract services are related to each other by semantic relationships, as explained below;
- *subject categories*, that organize abstract services into a standard available taxonomy (such as UNSPSC or NAICS) to provide a topic-driven access to the underlying services.

According to the distinction made above, the service ontology contains concrete services, abstract services and subject categories organized into three layers (called *Concrete*, *Abstract* and *Category* layer, respectively). This organization of *e*-Services is meant to support service provisioning properly adapted to changes in user context and quality conditions. Its main issue in the framework of the MAIS Project is to enhance finding of generic services (*abstract services*) describing the required functionalities that can be actually provided by several specific existing services (*concrete services*). Thus, abstract services are intended to shorten the way towards a variety of alternative concrete services that can be invoked. Context information and quality requirements further refine and filter the set of candidate concrete services. In addition, subject categories give the user a mechanism for an easy access to the underlying levels on the basis of standard topics. Concrete services are stored in UDDI Registry, that is properly extended (through the tModel element [2]) to contain service descriptions according to the service model proposed in Section 2.

3.1 Concrete layer construction

In [15] a methodology to compare services on the basis of their functional description is proposed in order to establish semantic relationships among them on the basis of their degree of similarity. The similarity between services is evaluated through the computation of coefficients obtained by comparing input/output parameters that services exchange during their execution (*Entity-based similarity analysis*) and operations that they are able to perform (*Functionality-based similarity analysis*) [16,17].

The similarity analysis is supported by a domain ontology used to annotate I/O parameters and operation names. In this ontology, terms are organized by means of weighted terminological relationships (synonymy, generalization/specialization relationships) both extracted from a *pre-existing, domain independent basic ontology* (e.g., WordNet) and supplied by a domain expert.

The domain ontology can be viewed as a graph $\langle \mathcal{N}, \mathcal{E} \rangle$, where $\mathcal{N}$ is the

set of nodes (i.e., terms) and $\mathcal{E}$ the set of edges (i.e., relationships between terms). Each edge is represented in the form $\langle (n_h, n_k), t, \sigma \rangle$, where $n_h \in \mathcal{N}$ is the source node of the relationship, $n_k \in \mathcal{N}$ is the destination node, t is the kind of relationship and $\sigma \in (0, 1]$ is a weight associated to that kind of relationship. Different kinds of relationships present different implications for similarity; in particular, we have $\sigma_{SYN} > \sigma_{BT/NT}$, since synonymy expresses a higher semantic connection between terms than the other kind of relationship. In our experimentation, $\sigma_{SYN} = 1$ and $\sigma_{BT/NT} = 0.8$. Two terms (nodes) can be related by a chain of relationships: we call *path* of length l between two nodes $n, n' \in \mathcal{N}$ a finite ordered sequence of edges $\langle e_1, e_2, \dots e_l \rangle$ (with $e_1, e_2, \dots e_l \in \mathcal{E}$), where the source node of e_1 is n and the destination node of e_l is n', denoted with $n \rightarrow^l n'$; the length l is the number of relationships in $n \rightarrow^l n'$. The *strength* of $n \rightarrow^l n'$ is the value of the function $\tau : Paths(\langle \mathcal{N}, \mathcal{E} \rangle) \rightarrow (0, 1]$ that associates to $\rightarrow^l$ the product of the weights of all the relationships belonging to it, where $Paths(\langle \mathcal{N}, \mathcal{E} \rangle)$ is the set of all the paths on the ontology.

Definition (Name Affinity Coefficient) The Name Affinity Coefficient of two terms n_i and $n_j \in \mathcal{N}$, denoted by $NA(n_i, n_j)$, measures the strength of path between them in the domain ontology, computed as follows:

$$NA(n_i, n_j) = \begin{cases} 1 & if\ n_i = n_j \\ \tau(\rightarrow^l) & if\ n_i \neq n_j\ \wedge\ n_i \rightarrow^l n_j, l \geq 1 \\ 0 & otherwise \end{cases} \quad (1)$$

Between two nodes in the ontology there can exist more than one path and in this case the path with the highest strength is chosen. We say that n_i and n_j have *name affinity* ($n_i \sim n_j$) if $NA(n_i, n_j) \geq \alpha$, where $\alpha > 0$ is a given threshold imposed to filter terms with high degree of affinity.

We group together services that have the same or similar I/O parameters or have the same or similar operations; given the domain ontology, service similarity is evaluated on the basis of I/O parameters and operation names contained in the descriptors associated with services. For this purpose, we introduce a set of *similarity coefficients* for services.

Definition (Entity-based similarity coefficient) Given two services S_i and S_j, we denote with $IN(S_i)$ and $IN(S_j)$ (resp., $OUT(S_i)$ and $OUT(S_j)$) the sets of input parameter names (resp., output parameter names) of S_i and S_j. The *Entity-based similarity coefficient* of S_i and S_j, denoted by $ESim(S_i, S_j)$, is the measure of affinity between names of I/O parameters of S_i and S_j, considered in their totality, that is

$$ESim(S_i, S_j) = \frac{2 \cdot A_{tot}(IN(S_i), IN(S_j))}{\mid IN(S_i) \mid + \mid IN(S_j) \mid} + \frac{2 \cdot A_{tot}(OUT(S_i), OUT(S_j))}{\mid OUT(S_i) \mid + \mid OUT(S_j) \mid} \quad (2)$$

where $A_{tot}(IN(S_i), IN(S_j))$ (respectively, $A_{tot}(OUT(S_i), OUT(S_j))$) denotes the total value of affinity between the pairs of input (respectively, output)

parameters in S_i and S_j, and $|\ \ |$ denotes the cardinality of a given set. A_{tot} is obtained by summing up the values of name affinity coefficients for all the pairs of input/output parameters that have name affinity in the domain ontology. Furthermore, we require that each parameter name participates at most in one pair for the A_{tot} evaluation. $ESim(S_i, S_j)$ assumes value 0 when no pairs of I/O parameters with name affinity are found, one from S_i and one from S_j, while it is 2 when S_i and S_j have the same input and output parameters.

Definition (Functionality-based similarity coefficient) Given two services S_i and S_j, we consider each pair of operations op_i and op_j, one from S_i and one from S_j. We denote with $IN(op_i)$ and $IN(op_j)$ (resp., $OUT(op_i)$ and $OUT(op_j)$) the set of input parameters (resp., output parameters) of op_i and op_j. The *operation similarity coefficient* between op_i and op_j is computed as follows

$$OpSim(op_i, op_j) = NA(op_i, op_j) + \frac{2 \cdot A_{tot}(IN(op_i), IN(op_j))}{\mid IN(op_i) \mid + \mid IN(op_j) \mid} + \frac{2 \cdot A_{tot}(OUT(op_i), OUT(op_j))}{\mid OUT(op_i) \mid + \mid OUT(op_j) \mid} \quad (3)$$

We note that $OpSim(op_i, op_j) \in [0, 3]$ since it is the sum of three elements in the range [0,1]. It is 0 when there is no affinity between operation names and I/O parameter names of two operations, while it is 1 when two operations have the same name and the same I/O parameters. We say that two operations op_i and op_j are similar, denoted by $op_i \sim op_j$, if the following conditions hold: (i) $OpSim(op_i, op_j) \geq \gamma$, where $\gamma > 0$ is a similarity threshold set by the domain expert; (ii) each of the three terms on the right hand side of (3) is greater than 0.

The *Functionality-based similarity coefficient* of two services S_i and S_j, denoted by $FSim(S_i, S_j)$, is the measures of similarity of their operations, computed as follows

$$FSim(S_i, S_j) = \frac{2 \cdot \sum_{h,k} OpSim(op_h, op_k)}{\mid OP(S_i) \mid + \mid OP(S_j) \mid} \quad (4)$$

where op_h (respectively op_k) denotes an operation of S_i (respectively of S_j), $op_h \sim op_k$ holds and $\mid OP(S_i) \mid$ and $\mid OP(S_j) \mid$ denote the number of operations of S_i and S_j, respectively. Note that $FSim(S_i, S_j) \in [0, 3]$, since each term $OpSim(op_h, op_k) \in [0, 3]$.

Finally, the *Global similarity coefficient* of two services S_i and S_j, denoted by $GSim(S_i, S_j)$, is the measure of their level of overall similarity computed as the weighted sum of the Entity-based and Functionality-based similarity coefficients as follows:

$$GSim(S_i, S_j) = w_1 \cdot NormESim(S_i, S_j) + w_2 \cdot NormFSim(S_i, S_j) \quad (5)$$

where $GSim() \in [0,1]$ since $NormESim()$ and $NormFSim()$ are respectively the values of $ESim()$ and $FSim()$ normalized to the range $[0,1]$; weights w_1 and w_2, with w_1, $w_2 \in [0,1]$ and $w_1 + w_2 = 1$, are introduced to assess the relevance of each kind of similarity in computing the Global similarity coefficient. The use of weights in $GSim(S_i, S_j)$ is motivated by the need of flexible comparison strategies. For instance, to state that the Entity-based similarity and Functionality-based similarity have the same relevance, we choose $w_1 = w_2 = 0.5$.

If this value is equal or greater than a threshold given by experimental results, S_i and S_j are grouped in the same set or *cluster*. Given a threshold ϕ, a cluster Cl is a set of concrete services $\Sigma = \{S_j\}$, $j = 1, ..n$, where $GSim(S_j, S_k) \geq \phi$, for each $S_j, S_k \in \Sigma$ and $k \neq j$. Clustering is performed following a traditional hierarchical procedure [18].

3.2 Abstract layer construction

In the *Abstract layer*, abstract services are chosen to represent the functionalities of concrete services in each cluster. The abstract service interface is obtained with an integration process performed on the interfaces of the concrete ones belonging to the cluster corresponding to the abstract service. In particular, the set of defined functionalities is "minimal", that is, operations in the abstract service interface are only those common to all the services in the cluster. The designer can force additional operations considered relevant because belonging to most services of the cluster.

Moreover, *mapping rules* are defined to relate operations in the abstract service with the corresponding ones in the concrete services. For each operation of the abstract service a table representing the mapping rules is built. Such a table has three columns: operation name, input data names and output data names. The first row is populated with the information about the abstract service, whereas the following ones represent the information about the concrete services in the corresponding cluster. In this way we can compare the names used in the abstract service to identify both the operation and the exchanged data with respect to the names used in the concrete services.

An `association` link is maintained between each abstract service and the corresponding cluster. Two kinds of semantic relationships are added between abstract services:

- `is-a`, that holds when an abstract service offers at least the same functionalities of another one; we say that, given two abstract services S_{a_1} and S_{a_2}, (S_{a_1} `is-a` S_{a_2}) if and only if for each operation op_2 in S_{a_2} there exists an operation op_1 in S_{a_1} where $NA(op_1, op_2) \geq \alpha$, the set of op_1 outputs includes the set of op_2 outputs and the set of op_2 inputs includes the set of op_1 inputs;
- `is-composed-of`, that is obtained when an abstract service can be viewed as the composition of other abstract services; we say that, given abstract

services S_{a_1} and $S_{a_2} \ldots S_{a_n}$, (S_{a_1} `is-composed-of` $\{S_{a_2} \ldots S_{a_n}\}$) if and only if for each operation op_1 in S_{a_1} there exists an operation op_2 in S_{a_2} or S_{a_3} or ... or S_{a_n} such that $NA(op_1, op_2) \geq \alpha$, the set of op_2 outputs includes the set of op_1 outputs and the set of op_1 inputs includes the set of op_2 inputs; the operations in $S_{a_2} \ldots S_{a_n}$ constitute a partition of the set of operations in S_{a_1}.

At the moment, semantic relationships are derived with the support of the domain expert; future efforts will be directed to the semi-automatic derivation of them. Semantic relationships will be properly used during the discovery process, as shown in Section 4.

3.3 Category layer construction

In the *Category layer*, a standard classification of services is considered to give users a topic-driven access to the underlying services. We considered the UNSPSC classification [19] to organize services in a commonly accepted service taxonomy in UDDI Registry, but this taxonomy is not the only existing effort for service classification and the choice of a different one could be made, since our approach is general enough from this point of view. Each abstract service is associated to one or more service categories in the taxonomy, maintaining an `association` link between them.

During the construction of the three-layer service ontology the designer is supported by a software tool environment, ARTEMIS [20], that evaluates by means of semi-automatic techniques the coefficients we introduced. ARTEMIS offers a value-added semi-automatic system that facilitates the ontology construction in a dynamic, highly evolving environment.

3.4 Running example

As application scenario, we consider a traveler who has to travel for business to Riga. His functional requirements concern the reservation of a flight from Milan to Riga and a hotel booking in the city; suppose that he does not want to spend too much for the air transfer and the hotel accomodation, so he requires a low-cost flight and a Latvian hotel that has no more than three stars. Finally, he would prefer to reserve the flight and book the hotel room by means of his credit card.

In this scenario, we consider a portion of three-layer service ontology, where flight reservation services, low-cost flight reservation services and hotel booking services are organized as shown in Figure 3. In particular, we suppose that there exist services that offer only flight reservation facilities, services that offer only hotel booking facilities and air travel services that offer both of them. In the portion of service ontology shown in Figure 3, ten concrete services are grouped into four clusters on the basis of their functional features and four abstract services representing the clusters are organized by means

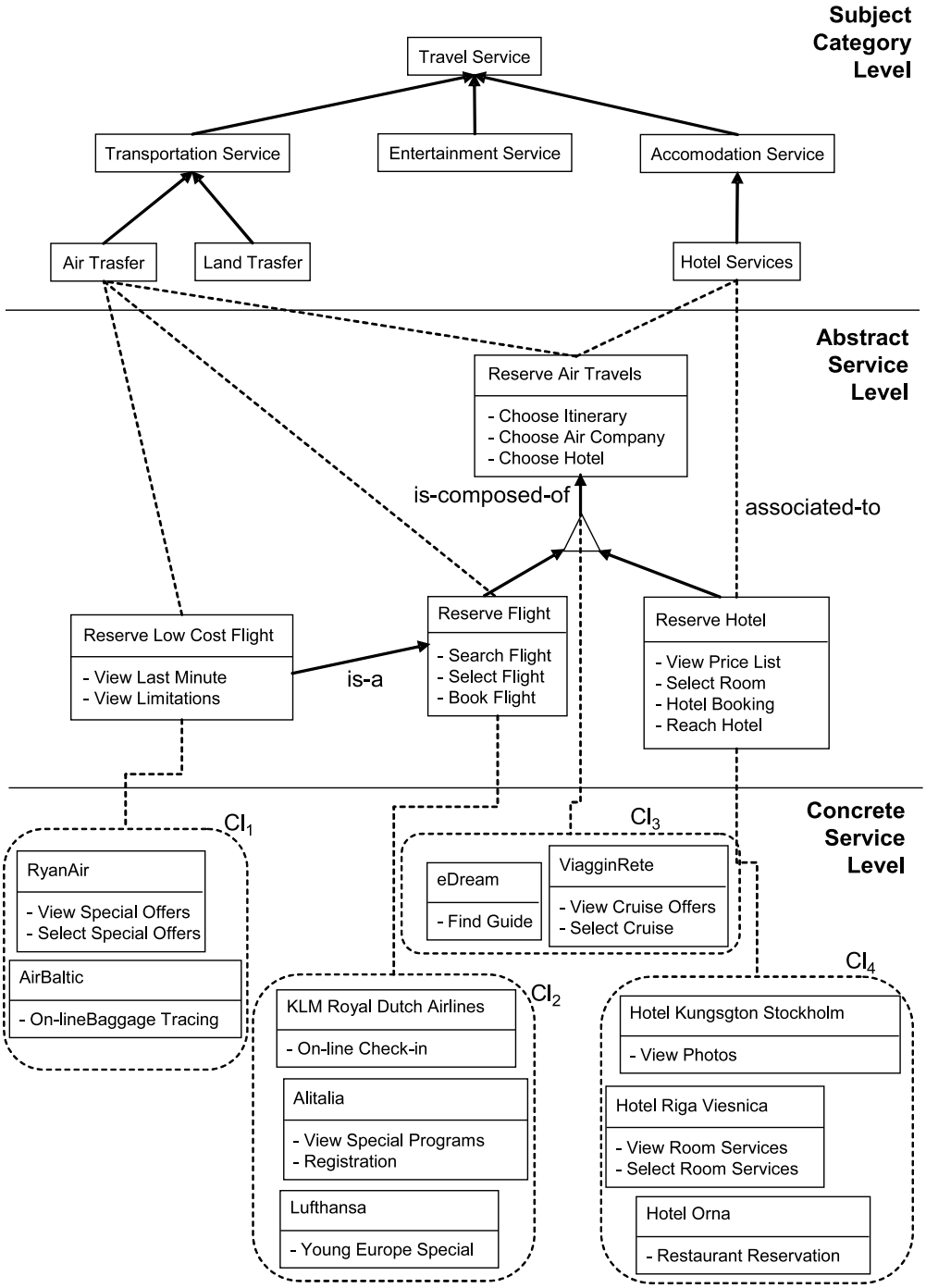

Fig. 3. A portion of three-layer service ontology for the running example.

of semantic relationships: `Reserve Low Cost Flight` abstract service is generalized by the `Reserve Flight` one (`is-a` relationship), while `Reserve Air Travels` service groups functionalities both of `Reserve Flight` service and `Reserve Hotel` service (`is-composed-of` relationship). These abstract ser-

vices are related to `Air Transfer` and `Hotel Services` standard categories. In the figure, we have omitted abstract functionalities inherited by means of semantic relationships. So, for example, `Search Flight` in the `Reserve Flight` service is also an operation of `Reserve Low Cost Flight` and `Hotel Booking` in the `Reserve Hotel` service is also an operation of `Reserve Air Travels`. Similarly, concrete operations integrated into the abstract ones are not represented for space restriction in the concrete services.

In the following section, our aim is to show how we can exploit organization of concrete and abstract services offered by the service ontology to support the user during the service discovery process.

We can summarize the user functional, quality and contextual requirements as follows:

Flight Reservation request		**Hotel Booking request**	
`Category`	flight, reservation	`Category`	hotel, booking
`Operation`	Reserve Flight	`Operation`	Book Hotel
`Inputs`	creditCardNumber	`Outputs`	booking confirmation
`Outputs`	flightElectronicTicket	`Context`	
`Context`		`Location`	Riga
`Location`	Milan, Riga	`Quality`	
`Quality`		`Parameter`	HotelStars
`Parameter`	FlightCost	`Constraints`	LessThanEqual(3)
`Constraints`	LessThanEqual(500)	`Parameter`	ResponseTime
`Units`	Euro	`Constraints`	1
		`Units`	Day

4 *E*-Service discovery

In this section, we show how to use functional matching techniques, user quality requirements, context information and the service ontology structure to enhance service discovery. Hereafter, we will use the notation Σ_{cand} to indicate the set of candidate abstract services to be presented to the user and with $\bar{\Sigma}_{cand}$ the set of corresponding concrete ones. Note that we populate $\bar{\Sigma}_{cand}$ starting from Σ_{cand} by means of `association` relationships among abstract and concrete services.

Functional, context and quality comparison are used to perform several kinds of searching modalities that are made available by the proposed plat-

form, extending functionalities of traditional UDDI Registry: we distinguish among *search by category*, *search by functionality* and *full search*. Different searching modalities can be combined to enhance service discovery and to make it more precise.

4.1 Search by category

This kind of search can be performed by the human user by browsing the subject category taxonomy at the top layer of the ontology and is very close to UDDI Registry searching facilities. Subject categories allow a preliminary filtering of candidate abstract services, that can be further refined by using other search modalities.

The result of this phase is the set Σ_{cand} of candidate abstract services that are related to the selected categories, ordered with respect to some predefined criteria:

- abstract services can be ordered in a decreasing way with respect to the number of selected categories they belong to;
- if there exists an `is-a` relationship between abstract services, the more general ones are presented before the other ones;
- composite abstract services, i.e., related to other services by means of the `is-composed-of` relationship, are presented before the component ones.

Example. Suppose that user, browsing the standard taxonomy, selects the Travel Service > Transportation Service > Air Transfer and Travel Service > Accomodation Service > Hotel Service categories. All abstract services related to them are proposed, in this case $\Sigma_{cond} = \{$`Reserve Air Travels`, `Reserve Hotel`, `Reserve Flight`, `Reserve Low Cost Flight`$\}$, displayed in this order since (i) the first one (`Reserve Air Travels`) belongs to both the selected subject categories and is composed by `Reserve Flight` and `Reserve Hotel` services, (ii) the abstract service `Reserve Flight` is more general than the abstract service `Reserve Low Cost Flight`. The order of `Reserve Hotel` service with respect to `Reserve Flight` and `Reserve Low Cost Flight` is not relevant.

4.2 Search by functionality

This search modality is performed on the abstract layer and returns an ordered set of candidate abstract services that are functionally similar to the requested one. Requested service S_R is expressed as the set of desired operations and I/O entities, with the support of the domain ontology introduced in Section 3. Similarity coefficients proposed in Section 3 are then exploited to evaluate the functional similarity between S_R and each abstract service S_{a_k}. In particular, we consider the *global functional similarity coefficient* between S_R and S_{a_k}, that we rename as $GSim_R(S_{a_k})$ and it is computed as shown in

the equation (5). Abstract services in Σ_{cand} are ordered on the basis of $GSim_R$ values.

Example. If we consider operations provided by the `Reserve Hotel` abstract service and the operations required by the user, we have that the `Book Hotel` required functionality matches with `Hotel Booking` operation, but the `Reserve Flight` requested functionality does not match with any operations provided by the `Reserve Hotel` service. In a similar manner, we can evaluate this matching for the other abstract services with the required operations and we obtain the following similarity values:

Abstract Service S_{a_k}	$GSim_R(S_{a_k})$ **value**
`Reserve Air Travels`	0.85
`Reserve Flight`	0.55
`Reserve Low Cost Flight`	0.55
`Reserve Hotel`	0.55

The abstract services are ordered on the basis of $GSim_R$ values and, in the case of very close values, exploiting the semantic relationships between them. Also in this case, `Reserve Flight` is proposed before the `Reserve Low Cost Flight` one, since it is more general; the order between these two services and the `Reserve Hotel` service is not relevant because no semantic relationships exist between them.

The user can choose, for example, the first proposed abstract service (`Reserve Air Travels`) and only the concrete services that belong to the corresponding cluster are presented to him (in our example, `eDream` and `ViagginRete` concrete services).

4.3 Full search

This search modality starts from a functional comparison between requested and offered abstract services. Context information and quality requirements are then used to further reduce the set $\bar{\Sigma}_{cand}$ of candidate concrete services proposed to the user. In fact, the analysis of the functional aspects of an *e*-Service is only one of the required steps to realize a useful service discovery mechanism. A potential user searches for a service that not only satisfies his requirements in terms of provided functionalities, but also guarantees a given quality of service level and respects the context in which the service will be used.

4.3.1 Context comparison

Focus of this phase is to exclude from the $\bar{\Sigma}_{cand}$ the services which do not satisfy the context constraints. According to the context definition discussed in Section 2, this analysis has to take into account the time-zone, the location and the channel constraints expressed by the user. In particular, an *"inclusion"* property is introduced for each of these constraints. So, given the requested e-Service S_R and a generic concrete e-Service S_i belonging to the $\bar{\Sigma}_{cand}$, S_i still remains in $\bar{\Sigma}_{cand}$ after the context comparison if $Context(S_R)$ is included in $Context(S_i)$ where:

$$\begin{aligned} Context(S_R) \subseteq Context(S_i) \quad iff \quad & TimeZone(S_R) \subseteq TimeZone(S_i) \wedge \\ & Location(S_R) \subseteq Location(S_i) \wedge \\ & Channel(S_R) \subseteq Channel(S_i) \end{aligned} \tag{6}$$

The *TimeZone* inclusion is defined in a natural way imposing that S_i is up during the interval of time defined by the S_R.

About the *Location*, the inclusion has to take into account the different levels in which the location is defined. In this way, S_i has to cover at least the area required by S_R. Thus we define:

$$Location(S_R) \subseteq Location(S_i) \quad iff \quad \begin{cases} Location(S_R) = \emptyset \\ Location(S_R) = Location(S_i) \wedge \\ \qquad\qquad C_{S_R} \subseteq C_{S_i} \end{cases} \tag{7}$$

$$C_{S_R} \subseteq C_{S_i} \quad iff \quad \begin{cases} C_{S_R} = \emptyset \\ C_{S_R} = C_{S_i} \wedge T_{S_R} \subseteq T_{S_i} \end{cases} \tag{8}$$

$$T_{S_R} \subseteq T_{S_i} \quad iff \quad \begin{cases} T_{S_R} = \emptyset \\ T_{S_R} = T_{S_i} \wedge D_{S_R} \subseteq D_{S_i} \end{cases} \tag{9}$$

$$D_{S_R} \subseteq D_{S_i} \quad iff \quad \begin{cases} G_{S_R} = \emptyset \\ G_{S_R} \simeq G_{S_i} \end{cases} \tag{10}$$

where:

- C_{S_i} is the Country of S_i
- T_{S_i} is the Town of S_i
- D_{S_i} is the District of S_i
- G_{S_i} is the GeoPosition of S_i

As it can be noted, the comparison between the GeoPosition does not require a strict equivalence. According to the wanted precision and the tolerance of the GPS instruments, the equivalence can be re-defined.

With respect to the *Channel*, an inclusion property definition similar to that defined for the other two context dimensions is not possible. On the contrary, with the help of the domain expert, it is more useful to identify a set of compatibility rules that state when a type of channel is compatible with another one. In this way $Channel(S_R) \subseteq Channel(S_i)$ if the two channels either are the same or, according to the compatibility rules, are considered compatible. The rules are defined according to the knowledge which characterizes the network and device community. For example, we can state that a PC connected to a wired LAN can be considered compatible with respect to a PC connected to a wireless LAN. In this case, from a functional standpoint, a compatibility relationship exists, but the user could prefer one of the two channels with respect to the available bandwidth. This kind of analysis is not performed in this step, but it is an investigation area of the quality comparison discussed in the following paragraph.

Example. If we are looking for a `Reserve Low Cost Flight` to Riga, the previous phase returns all the concrete services belonging to the Cl_1 cluster, i.e., $\bar{\Sigma}_{cand} = \{\texttt{RyanAir}, \texttt{AirBaltic}\}$. Now, let us suppose that the `RyanAir` is specialized in flights among the Western European cities, whereas the AirBaltic, as the name suggests, is specialized in the Baltic region.

Supposing that an algorithm able to process and verify the relationship among the worldwide cities is available, the relationship (9) is not satisfied for the `RyanAir` service, since Riga belongs to the Eastern Europe. In this way, the $\bar{\Sigma}_{cand}$ becames $\bar{\Sigma}_{cand} = \{\texttt{AirBaltic}\}$.

4.3.2 Quality comparison

The quality model presented in the Section 2 allows the user to define, through the QoE, which is the requested quality, according to the same quality parameter set defined for the QoS and taking also into account the effects of the channels. Thus the S_R can define the expected quality ($\text{QoE}(S_R)$) in two ways, i.e., dependent or independent from a particular channel.

In the first case, starting from the QoS defined for a $S_i \in \bar{\Sigma}_{cand}$, the $\text{QoE}(S_i)$ is computed only about the channels defined in the S_R context. Obviously, at this stage all the S_i can support the requested channel, since the context comparison has already verified this restriction over $\bar{\Sigma}_{cand}$. Once $\text{QoE}(S_i)$ is computed, it is compared with the quality values requested by the user $\text{QoE}(S_R)$ in order to verify that all the quality parameter values satisfy the requested quality parameter value.

In the second case, if S_R does not specify any channels, it means that the $\text{QoE}(S_R)$ should be valid for at least one of the S_i available channels. So, the $\text{QoE}(S_i)$ is computed for all the channels in $\text{Channel}(S_i)$ and the obtained values are compared with the $\text{QoE}(S_R)$. In this way, for each S_i we could have a set of channels which satisfy the quality constraints. It is worth nothing that this kind of selection adds a new leverage about the adaptivity during

the service invocation. In fact the service, in case of quality decreasing in the current channel, can switch among the other available channels in order to maintain the quality level above the requested one.

Example. The requested hotel booking service must provide a response at least within one day, but no specific channel is requested. For this matter, we have to find, for each service in the $\bar{\Sigma}_{cand}$ = {`HotelKungsgton Stockholm`, `HotelRigaVienisca`, `HotelOrna`} (we are supposing that all of these services provide the requested functionalities), which are the channels which satisfy the QoE(S_R). Actually, after the context comparison, the `Hotel Kungsgton Stockholm` is already deleted from this set, thus the quality comparison is performed only for the services $\bar{\Sigma}_{cand}$ = {`HotelRigaVienisca`, `HotelOrna`}. Now, let us suppose that both services can be invoked through the Web and that only the `Hotel Orna` also via GSM SmartPhone, with the following QoE:

Service	**Channel**	**Response Time**
`Hotel Orna`	Web	2 days
`Hotel Orna`	GSM	6 hour
`Hotel Riga Vienisca`	Web	10 sec

In this case, the `Hotel Orna` can be invoked only using the SmartPhone, since the Web channel is affected by a lot of manual back office work which delays the response. Otherwise, the only available channel for `Hotel Riga Vienisca` service can be used.

4.4 Exploiting semantic relationships for service discovery

Semantic relationships among abstract services (`is-a` and `is-composed-of`) can be exploited to support the user in finding services that better fit his requirements. In fact, if a proposed service does not satisfy user requirements, semantic relationships can be used to propose other abstract services (and corresponding concrete ones) related to the previously selected one. For example, the functionality required by the user could be performed not only by a single service, but also by a composition of services.

4.4.1 Generalization/specialization

Suppose that no concrete services in the selected cluster satisfy user quality requirements or context constraints: the abstract service that represents the cluster is considered and, if other abstract services exist that specialize it, their corresponding clusters of concrete services are presented to the user. In fact, if an abstract service S_{a_k} specializes another abstract service S_{a_j}, then it provides at least the same functionalities, as exposed in Section 3.

Thus, the set of candidate abstract services is extended as follows:

$$\Sigma'_{cand} = \Sigma_{cand} \cup \{Sa_k \mid (Sa_k \texttt{ is} - \texttt{a } Sa_j),\ \forall Sa_k \in \Sigma_{cand}\} \tag{11}$$

Example. If concrete services associated to the `Reserve Flight` abstract one do not match user cost requirements, the `is-a` relationship could be exploited and the set of concrete services represented by the `Reserve Low Cost Flight` abstract one are proposed to the user, that is, $\Sigma_{cand} = \{$`Reserve Flight`$\}$, $\Sigma'_{cand} = \{$`Reserve Flight`, `Reserve Low Cost Flight`$\}$, $\bar{\Sigma}'_{cand} = \{$`RyanAir`, `AirBaltic`$\}$. Note that $\bar{\Sigma}'_{cand}$ contains only concrete services belonging to Cl_1, since concrete ones in Cl_2 did not respect quality and/or context constraints.

4.4.2 Composition

The same considerations made above can be repeated in the case of composition relationship, where S_{a_j} is the composite service and S_{a_k} the union of component services: the component services can be considered and proposed together to the user. The set of candidate abstract services becomes:

$$\Sigma'_{cand} = \Sigma_{cand} \cup \{Sa_h \mid Sa_h \in \sigma \wedge (Sa_j \texttt{ is} - \texttt{composed} - \texttt{of } \sigma), \quad \forall Sa_j \in \Sigma_{cand}\} \tag{12}$$

Example. In the running example, if any concrete service corresponding to the `Reserve Air Travels` abstract service does not satisfy the user quality requirements or context information, then the `is-composed-of` relationship is exploited to propose to the user the concrete services associated to the `Reserve Flight` and the `Reserve Hotel` abstract ones. At this point the user must search for a flight and a hotel reservation by choosing separately a concrete service from the cluster related to the `Reserve Flight` service and a concrete service from the cluster associated to the `Reserve Hotel` one. In this case, $\Sigma_{cand} = \{$`Reserve Air Travels`$\}$, $\Sigma'_{cand} = \{$`Reserve Air Travels`, `Reserve Flight`, `Reserve Hotel`$\}$, $\bar{\Sigma}'_{cand} = \{$`Hotel Kungsgton Stockholm`, `Hotel Riga Viesnica`, `Hotel Orna`, `KLM Royal Dutch Airlines`, `Alitalia`, `Lufthansa`$\}$.

Application of rules proposed above can be iterated until the set Σ'_{cand} will stop changing. For example, we can extend the set of candidate services by exploiting the `is-composed-of` relationship between `Reserve Air Travels`, `Reserve Flight` and `Reserve Hotel` services and we can further extend the set by exploiting the `is-a` relationship between `Reserve Flight` and `Reserve Low Cost Flight` services.

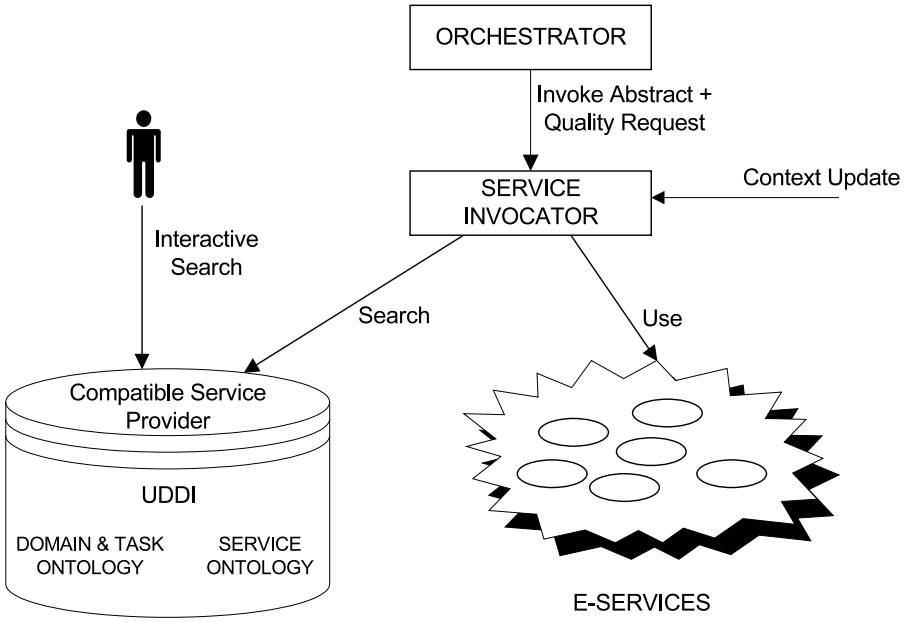

Fig. 4. MAIS architecture for service classification and discovery.

5 The architecture

The models and methods introduced are used within the architectural framework shown in Figure 5, which enables service discovery as described above. In particular, one of the main goals of the work is to propose an architecture that extends the UDDI Registry, maintaining a full compatibility with it. In such a way the user can either exploit the classical UDDI APIs or invoke the APIs provided by our architecture which provide the additional functionalities discussed in the present paper. The UDDI functionalities are also used to store the *e*-Service specifications according to the service model presented in Section 2. In fact, as described in [21], the UDDI items could refer not only to WSDL files, but also to generic specifications by means of the extension mechanism of tModels. At this time we are working on a XML-specification about the context and quality properties of an *e*-Service.

Starting from the data repository, the system relies on the ontologies, as highlighted in Sections 3 and 4: a domain-specific ontology, in which the terms used to semantically describe services are organized, and a service ontology, that organizes the published services as described in Section 3.

The functionalities of the architecture are exported through both an API and a Web application, which allows the user (human or application) to interact with the Compatible Service Provider (hereafter CSP). Through such functionalities the user can either (i) require a service publication or (ii) look for a service not only using the typical UDDI facilities, but also the new mechanisms described in this work. In the first case the CSP, according to the similarity functions described in Section 3, is in charge of correctly placing the *e*-Service inside the service ontology, whereas in the second case it is able to

identify a set of candidate services with respect to a given request.

The CSP can be invoked by a user, who wants to perform an interactive search (for example, starting with a *search by category* modality) or by an application to execute a process through the invocation of a set of *e*-Services in a given order. In particular, in this last case an *Orchestrator* splits the process execution into a sequence of service requests with quality constraints which are sent to the *Service Invocator*, a module that, after collecting context information, interacts with the CSP exploiting searching modalities presented in Section 4 to obtain references to the desired concrete services. Finally, these services can be invoked and used for the process execution.

6 Related work

Several approaches in literature address issues related to service description and modeling, service classification and matchmaking, on the basis of functional, contextual and quality-based comparison and use of ontologies to enhance service discovery. A large number of proposals attempt to maintain compatibility with existing standards in the service oriented architecture, in particular with SOAP, WSDL and UDDI.

The service directory holds an important role inside the Service Oriented Architecture and different proposals are available in order to provide models and mechanisms not only for the service publication, but also for a more efficient service discovery. UDDI Registry [2] is the most important and known implementation of a service directory. The main task of a UDDI Registry is to organize services with respect to three different aspects: who provides the service, which functionalities the service performs and how the service can be invoked. In this way, the UDDI Registry, besides a classical keyword-based search, can be browsed according to three different modalities, namely through White Pages, Yellow Pages and Green Pages. Several implementations of the UDDI Registry are now available [2], but extensions have been proposed due to the lack of a semantic-based classification and search of services.

In our approach, the extensions to UDDI use a service ontology, with abstract and concrete services, and semantic context and quality information about services.

About the service modeling proposals, WSDL is a de-facto standard for service functionality description, whereas OWL-S [22] is meant to provide a description of everything a service can do by combining a *Service Profile* ("what the service does" enriched with non functional aspects such as quality criteria), a *Process Model* ("how the service works") and a *Service Grounding* (related to implementation details).

In our approach, in addition to describe simple services, we focus on providing a general service ontology, which allows relating services to each other

[2] See `uddi.microsoft.com`, `uddi.ibm.com` and `uddi.sap.com`.

during classification and retrieval.

Quality of service issues are also addressed by languages as WSLA [23] or WSOL [24,25], which identifies the QoS parameters deemed useful for service providers to characterize services; in some sense, [26] integrates this proposal to extend service discovery based on QoS-related information other than on interfaces. [27] proposes a mechanism enabling the evaluation of the overall QoS of a composite service, i.e., a service obtained by composing several distinct services, once a description of the QoS parameters of the component services is given.

In this paper, we associate quality levels as a first class item in the description of services. We also propose criteria to select services on the basis of functional, contextual and quality aspects in the service request.

About profiles and context information, we can mention the WWRF (Wireless World Research Forum) [28], the Cameleon Project [29] and the CC/PP (Composite Capabilities/Preferences Profile) initiative of the W3C Device Independence Working Group [30]. The UWA Project studied and proposed *customization rules* [12] to describe and constrain the adaptability of ubiquitous Web applications. Fensel et al. [31] suggest the use of semantic annotation of service descriptions to allow automatic service discovery, composition, invocation and interoperation.

In particular, [32] proposes an extension of WSDL, that is called WSSP (Web Service Semantic Profile), to encode semantic information in WSDL by annotating I/O entities by means of domain ontologies written in RDF, DAML+OIL or OWL and by expressing constraints on inputs and outputs using an RDF-Rule Markup Language [33]. Semantic annotation is used both when a new service is registered into the UDDI Registry and when a service request is formulated. A semantic matchmaker is then used to perform matchmaking among a service request and the registered service descriptions. The matchmaker is inspired by LARKS [34] algorithm and the DAML-S matchmaker [35]. Firstly, the well-known information retrieval technique TD/IDF (Term Frequency Inverse Document Frequency) is used as a preliminar filter to locate candidate services in the UDDI Registry even when no semantic information has been provided. Then two other kinds of filters are applied, based on semantic description of services: (i) a *type filter*, that checks to see if the definitions of input and output parameters match by applying a set of subtype inferencing rules on the types of inputs and outputs (defined as classes in the domain ontologies); (ii) a *constraint filter*, that computes the logical implication among constraints by using polynomial subsumption algorithm for Horn clauses. These two filters are meant to verify if the required service can be replaced by a registered one (*plug-in match*) by evaluating I/O and constraint subsumption. Wang et al. [36] use both the semantics of the identifiers of WSDL descriptions and the structure of their operations, messages and data types to assess the similarity of WSDL files used to describe services. A semantic information-retrieval method is also used to identify and order the most similar service descriptions when only a textual description of

desired services is given.

In Section 4 we have discussed the algorithms for different types of search in our approach. Our focus is on selecting *e*-Services considering all interesting aspects in the request. In our work we propose to use a semantic approach in the classification of the services in the ontology and to be able to evaluate similarity between services.

Several efforts have been dedicated to the classification of services based on the use of ontologies in order to enhance their discovery. The actual use of the ontology during search is limited, mainly to allow scalability of the systems when a large number of services is considered. Puyal et al. [37] studied the selection and execution of remote services on mobile devices and proposed a system, REMOTE, where a service ontology stores information about the available services in a frame-based way simply using service categories and keywords to classify services in a taxonomy (only `is-a` relationships are taken into account). When a service request is formulated, a new concept (service) is created and positioned in the taxonomy with the support of a Description Logic reasoner. The child nodes of this new concept, once it is positioned in the ontology, are the services presented to the user. Mostéfaoui and Hirsbrunner [38] proposed an architecture for gathering and processing contextual information exploited to enhance service selection after the application of the discovery mechanism based on functional description of services. In [27] a service ontology is proposed to obtain an agreement by a community (for example, air transfer). Here, a service ontology specifies a domain, a set of synonyms to allow a flexible search for the domain and a set of service classes to define the properties of services, further specified by its attributes and operations. A service class represents a set of services providing the same capabilities (operations). The service ontology also specifies a service quality model that is used to describe non functional aspects.

As explained in the previous sections, our approach tries to capture existing efforts about service functional comparison grouping concrete services into clusters of similar ones and to generalize common functionalities provided by clustered concrete services by means of abstract service capabilities, as also suggested in [31], to shorten the way towards a variety of possible alternative concrete services that could be invoked and that are filtered in a second moment on the basis of quality and contextual issues. Semantic relationships among services are also derived and used for discovery purposes. This approach allows scalability in the search process, while compatibility with current SOA standards (in particular UDDI) ensures a certain level of interoperability.

7 Conclusions

Service discovery is one the most important emerging key aspects in Service Oriented Computing. The necessity for both a human and an application to find the more suitable *e*-Services with respect to a set of requirements is

discussed in this work. An *e*-Service model is introduced according to two different standpoints, in which both what the user wants (requester perspective) and what the provider offers (the provider perspective) are defined. Starting from this model, we suppose that all the available services are stored and organized inside a service ontology. This organization allows us to identify and maintain the possible relationships that could exist among services. Exploiting such an ontology, service discovery techniques are used to compare the requirements of the user specified using the requester perspective model against the functionalities provided by the services stored in the ontology and described according to the provider perspective. Retrieval techniques consider also the context of the request and the requested quality level to filter suitable services obtained after the previous functional retrieval phase.

A subset of the techniques introduced have been implemented and are now available in a tool called CSP (Compatible Service Provider), which uses ARTEMIS tool environment as the underlying ontology engine [20]. Future activities concentrate on the CSP completion and on the definition of an ad-hoc quality model within the MAIS Project. Behavioral aspects in the *e*-Service description will be considered by means of states and state transitions to further enhance functional comparison, while the implementation of an efficient and decidable reasoner that is able to manage constraints imposed on input and output parameters by pre- and post-conditions is also under development. Finally, experimentations of our approach in the touristic domain is currently addressed.

Acknowledgements

Part of this work has been supported by the Italian MIUR/MURST FIRB MAIS (Multichannel Adaptive Information Systems) Project.

References

[1] M. P. Papazoglou, D. Georgakopoulos, Service-Oriented Computing, special issue, guest editors introduction, Commun. ACM 46 (10) (2003), pag. 24–28.

[2] UDDI Technical White Paper, `www.uddi.org/pubs/lru_UDDI_Technical_Paper.pdf` (2001).

[3] T. Berners-Lee, J. Hendler, O. Lassila, The semantic Web, Scientific American 284 (5).

[4] R. Fillman, Semantic services, IEEE Internet Computing 7 (2003) 4–6.

[5] The MAIS Project Home Page, `www.mais-project.it`.

[6] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy, I. Trickvic, S. Zimek, Web Service Choreography Interface 1.0, `www.w3.org/TR/wsci` (August 2002).

[7] BPMI.org - Business Process Modeling Language, www.bpmi.org.

[8] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, S. Weerawarana, Business Process Execution Language for Web Services, version 1.0, IBM, www.ibm.com/developerworks/library/ws-bpel/ (July 2002).

[9] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, Web Services Description Language (WSDL) 1.1, World Wide Web Consortium (W3C), www.w3.org/TR/2001/NOTE-wsdl-20010315 (March 2001).

[10] C. Marchetti, B. Pernici, P. Plebani, A quality model for multichannel adaptive information, in: Proceedings of the 13th International World Wide Web Conference, Alternate track on Web Services, ACM Press, 2004, pp. 48–54.

[11] B. Pernici, P. Plebani, C. Batini, C. Cappiello, P. Missier, QoS in Multichannel IS: the MAIS Approach, in: Proceedings of the International Workshop on Web Quality (WQ'04) in conjunction with the ICWE 2004, Munich, Germany, 2004.

[12] UWA Consortium, UWA Web Site, www.uwaproject.org.

[13] Distributed Management Task Force Standards, www.dmtf.org/standards.

[14] A. van Moorsel, Metrics for the Internet Age: Quality of Experience and Quality of Business, Tech. rep., HP Labs, also in the Proceedings of the 5th Performability Workshop, September 16, 2001, Erlagen, Germany. (July 2001).

[15] D. Bianchini, V. De Antonellis, M. Melchiori, An ontology-based method for classifying and searching *e*-Services, in: Proc. Forum of First Int. Conf. on Service Oriented Computing (ICSOC 2003), Trento, Italy, 2003.

[16] V. De Antonellis, M. Melchiori, and P. Plebani, An Approach to Web Service compatibility in cooperative processes, in: Proc. IEEE SAINT2003 of Int. Workshop on Services Oriented Computing: Models, Architectures and Application (SOC2003), Orlando, Florida, USA, 2003.

[17] S. Castano and V. De Antonellis, A Framework for expressing Semantic Relationships between Multiple Information Systems for Cooperation, Information Systems, 23 (3-4) (1998) 253–277.

[18] B. Everitt, Cluster Analysis, Heinemann Educational Books Ltd, Social Science Research Council, 1974.

[19] ECCMA, UNiversal Standard Products and Services Classification (UNSPSC), www.eccma.org/.

[20] The ARTEMIS Project Home Page, www.ing.unibs.it/~deantone/interdata_tema3/Artemis/artemis.html.

[21] D. Ehnebuske, D. Rogers, C. von Riegen, UDDI Version 2.0 Data Structure Reference (June 2001), www.uddi.org/pubs/DataStructure-V2.03-Published-20020719.pdf.

[22] A. Ankolenkar, M. Burstein, G. Denker, J. R. Hobbs, O. Lassila, D. L. Martin, D. McDermott, D. McGuinness, S. A. McIllraith, M. Paolucci, T. R. Payne, B. Parsia, M. Sabou, E. Sirin, N. Srinivasan, K. Sycara, M. Solanki, OWL-S: Semantic Markup for Web Services, OWL-S 1.0 Draft Release, (November 2003).

[23] A. Keller, H. Ludwig, The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services, Technical Report RC22456(W0205-171), IBM Research Division, T.J. Watson Research Center (May 2002).

[24] V. Tosic, K. Patel, B. Pagurek, WSOL - Web Service Offerings Language, in: Web Services, E-Business and the Semantic Web, CAiSE 2002 International Workshop on Web Services, E-business, and the Semantic Web (WES 2002), Toronto, Canada, 2002.

[25] A. Mani, A. Magarajan, Understanding Quality of Service of your Web services, IBM Developer Works `www.ibm.com/developerworks/library/ws-quality.html` (January 2002).

[26] S. Ran, A Model for Web Services Discovery with QoS, in: ACM SIGecom Exchange, Vol. 1, ACM Press, New York, NY, USA, 2003, pp. 1–10.

[27] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, H. Chang, QoS-Aware Middleware for Web Services Composition, IEEE Transactions on Software Engineering 30 (5) (2004) 311–327.

[28] Book of visions 2001, `www.wireless-world-research.org` (2001).

[29] Cameleon Consortium, Cameleon Web Site, `giove.cnuce.cnr.it/cameleon.html`.

[30] W3C Consortium, Composite Capabilities/Preferences Profile, `www.w3.org/Mobile/CCPP/`.

[31] R. Lara, H. Lausen, S. Arroyo, J. de Bruijn, D. Fensel, Semantic Web Services: description requirements and current technologies, in: International Workshop on Electronic Commerce, Agents and Semantic Web Services, in conjunction with the Fifth International Conference on Electronic Commerce (ICEC 2003), Pittsburgh, PA, 2003.

[32] T. Kawamura, J.-A. D. Blasio, T. Hasegawa, M. Paolucci, K. Sycara, Preliminary Report of Public Experiment of Semantic Service Matchmaker with UDDI Business Registry, in: Proc. of First Int. Conf. on Service Oriented Computing (ICSOC 2003), Vol. LNCS 2910, Springer, Trento, Italy, 2003, pp. 208–224.

[33] The Rule Markup Initiative, `www.dfki.uni-kl.de/ruleml/`.

[34] K. Sycara, S. Widoff, M. Klusch, J. Lu, LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace, Autonomous Agents and Multi-Agent Systems 5 (2002) 173–203.

[35] M. Paolucci, T. Kawamura, T. Payne, K. Sycara, Semantic Matching of Web Services Capabilities, in: Proc. of the First Int. Semantic Web Conference (ISWC2002), IEEE, 2002, pp. 333–347.

[36] Y. Wang, E. Stroulia, Semantic Structure Matching for Assessing Web-Service Similarity, in: Proc. of First International Conference on Service-Oriented Computing (ICSOC2003), Vol. LNCS 2910, Springer, Trento, Italy, 2003, pp. 194–207.

[37] J. M. Puyal, E. Mena, A. Illarramendi, REMOTE: a Multiagent System to Select and Execute Remote Software Services in a Wireless Environment, in: Proc. of WWW 2003 Workshop on E-Services and the Semantic Web (ESSW2003), Budapest, Hungary, 2003.

[38] S. K. Mostéfaoui, B. Hirsbrunner, Enhancing Service Composition with Context Information, in: Proc. Of Fifth International Conference on Information and Web-based Applications & Services, Jakarta, Indonesia, 2003.